NAME

copyin – copy data from a user program to a driver buffer

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ddi.h>

int copyin(caddr_t userbuf, caddr_t driverbuf, size_t cn);
```

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI).

ARGUMENTS

userbuf User program source address from which data is transferred.

driverbuf Driver destination address to which data is transferred.

cn Number of bytes transferred.

DESCRIPTION

copyin( ) copies data from a user program source address to a driver buffer. The driver developer must ensure that adequate space is allocated for the destination address.

Addresses that are word-aligned are moved most efficiently. However, the driver developer is not obligated to ensure alignment. This function automatically finds the most efficient move according to address alignment.

RETURN VALUES

Under normal conditions a **0** is returned indicating a successful copy. Otherwise, a **-1** is returned if one of the following occurs:

- paging fault; the driver tried to access a page of memory for which it did not have read or write access
- invalid user address, such as a user area or stack area
- invalid address that would have resulted in data being copied into the user block

If a **-1** is returned to the caller, driver entry point routines should return **EFAULT**.

CONTEXT

copyin( ) can be called from user context only.

EXAMPLES

A driver **ioctl**(9E) routine (line 9) can be used to get or set device attributes or registers. In the **XX_GETREGS** condition (line 17), the driver copies the current device register values to a user data area (line 18). If the specified argument contains an invalid address, an error code is returned.

```
1 struct  device  {                    /* layout of physical device registers  */
2       int     control;               /* physical device control word  */
3       int     status;                /* physical device status word   */
4       short   recv_char;             /* receive character from device */
5       short   xmit_char;             /* transmit character to device  */
6 }; /* end device */
7
8 extern struct device xx_addr[];       /* phys. device regs. location */
  . . .
```

```
 9  xx_ioctl(dev, cmd, arg, mode, cred_p, rval_p)
10        dev_t   dev;
11        int     cmd, arg;
12            ...
13  {
14     register struct device *rp = &xx_addr[getminor(dev) >> 4];
15     switch (cmd)  {
16
17      case XX_SETREGS:                  /* copy device regs. to user program */
18          if (copyin((caddr_t)arg, (caddr_t)rp, sizeof(struct device)))
19              return(EFAULT);
21          break;
```

SEE ALSO **ioctl**(9E), **bcopy**(9F), **copyout**(9F), **ddi_copyin**(9F), **ddi_copyout**(9F), **uiomove**(9F).

Writing Device Drivers

NOTES Driver writers who intend to support layered ioctls in their **ioctl**(9E) routines should use **ddi_copyin**(9F) instead.

Driver defined locks should not be held across calls to this function.